15-10-2021

INSTALACIÓN Y USO MongoDB y Visual Studio Code

+información útil

Francisco Buiza Pérez

Contenido

1. ¿Qué es MongoDB?

MongoDB es un sistema gestor de base de datos NoSQL que guarda estructura de datos en documentos .json, haciendo que la integración de los datos en ciertas aplicaciones sea mucho más rápida.

Para trabajar con mongoDB, instalaremos Visual Studio Code.

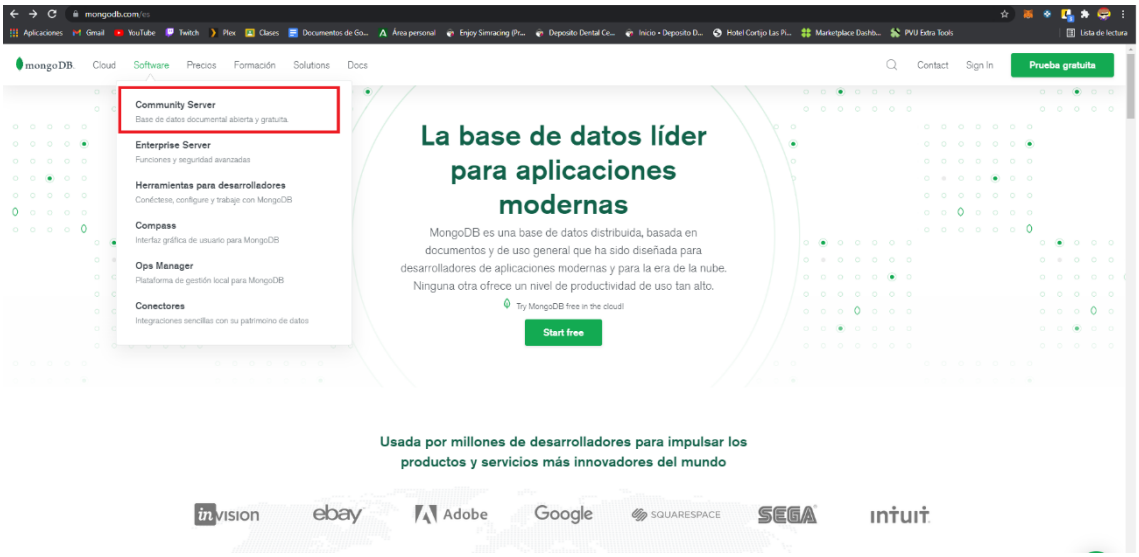
2. ¿Qué es Visual Studio Code?

Visual Studio Code es un editor de código desarrollado por Microsoft. Incluye muchas herramientas útiles, como soporte de depuración de código, resaltado de sintaxis, refactorización de código e incluso integración con Git. Sirve para una amplia gama de lenguajes (por no decir todos).

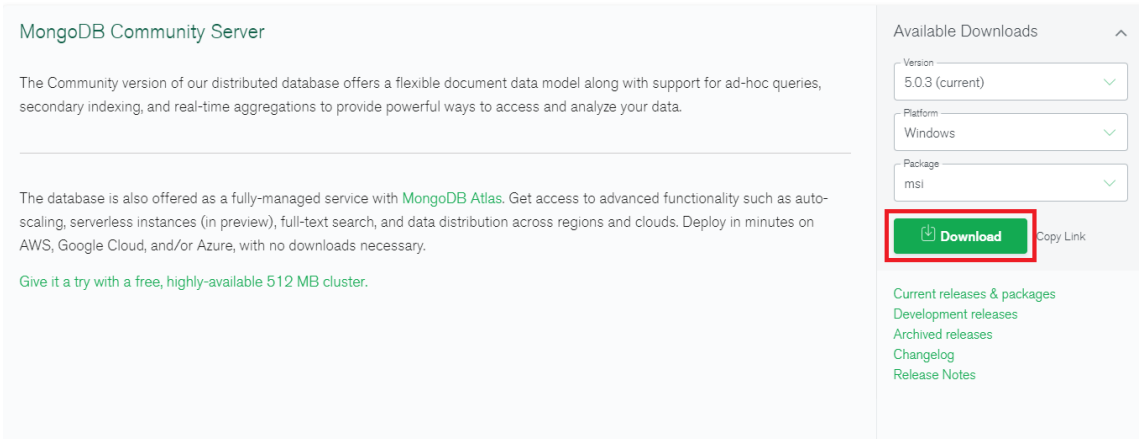
3. Instalación de MongoDB

Para instalar MongoDB, lo primero que tenemos que hacer es descargar el archivo instalador de la página oficial de Mongo.

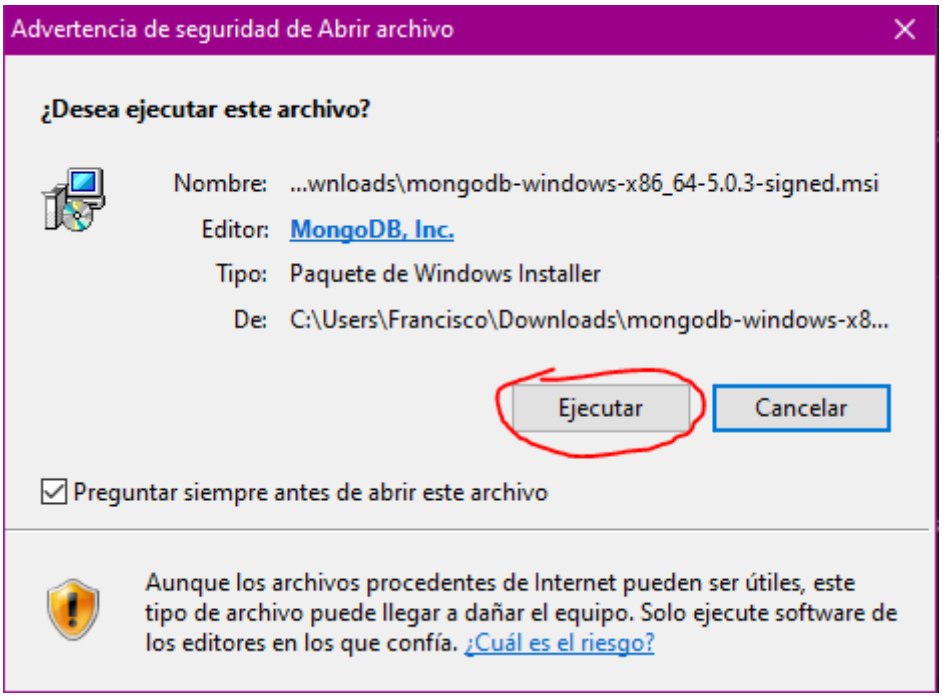
Una vez dentro, pulsaremos en el apartado de software, y accederemos a "Community Server", versión la cual es gratuita y abierta:



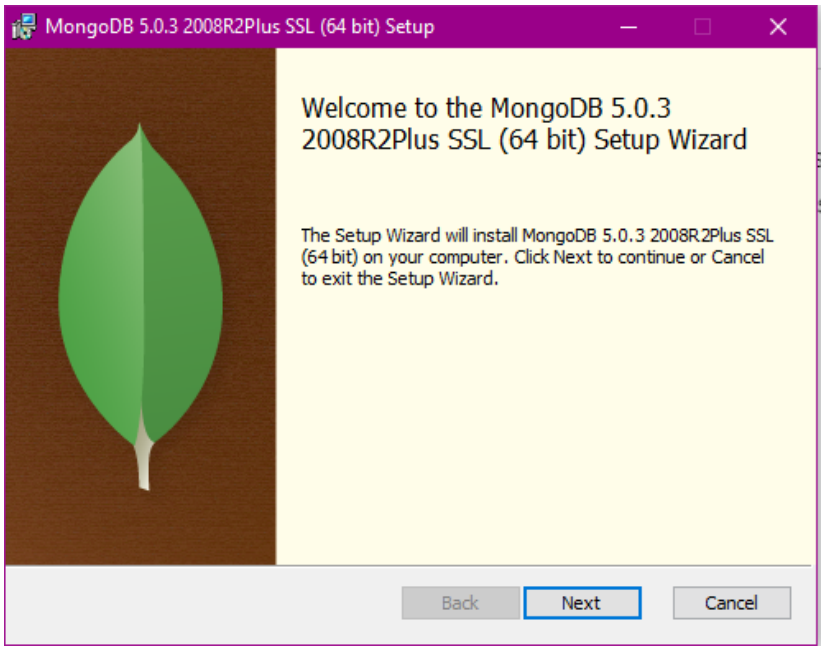
Una vez dentro, pulsaremos el botón de descarga y esta comenzará:



En descargas (o la carpeta en que se haya descargado el archivo) ejecutaremos el instalador de Mongo:

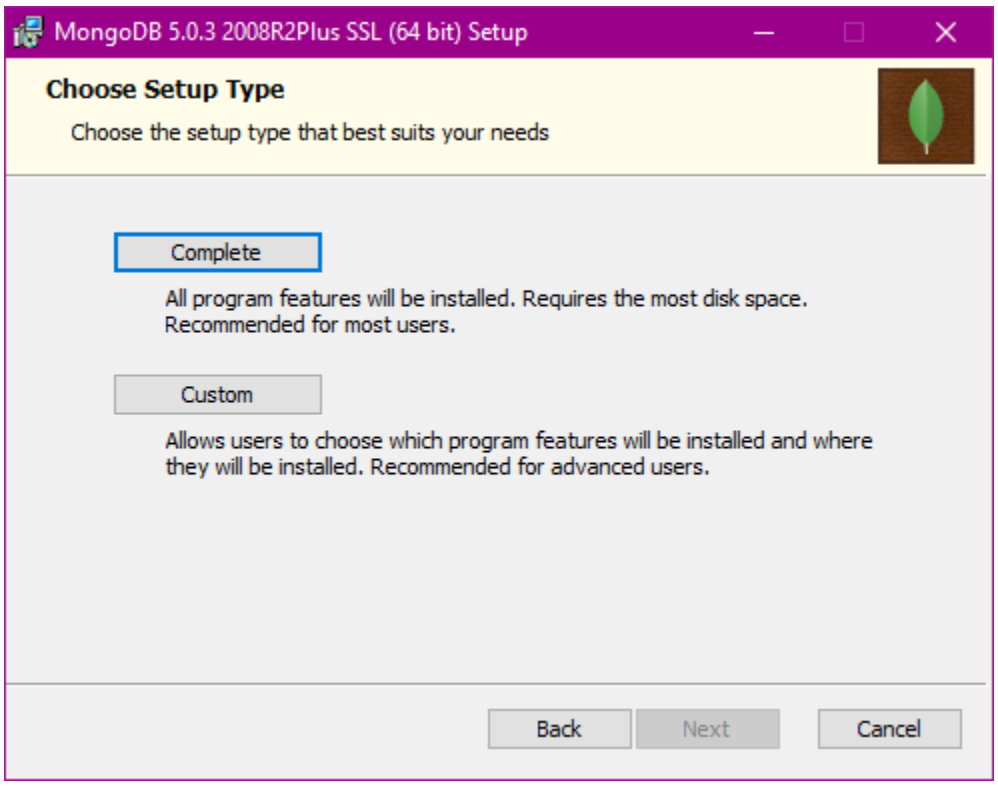


Ejecutamos el archivo.



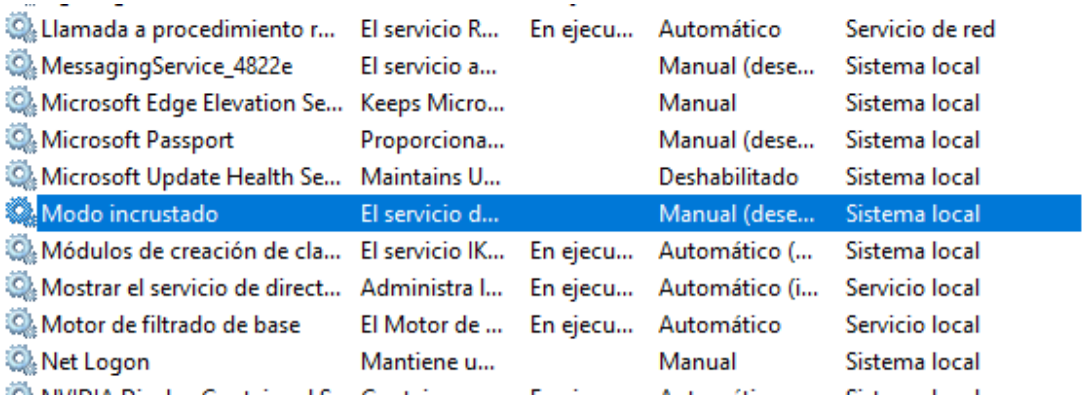
Pulsamos siguiente

Leemos y aceptamos los términos

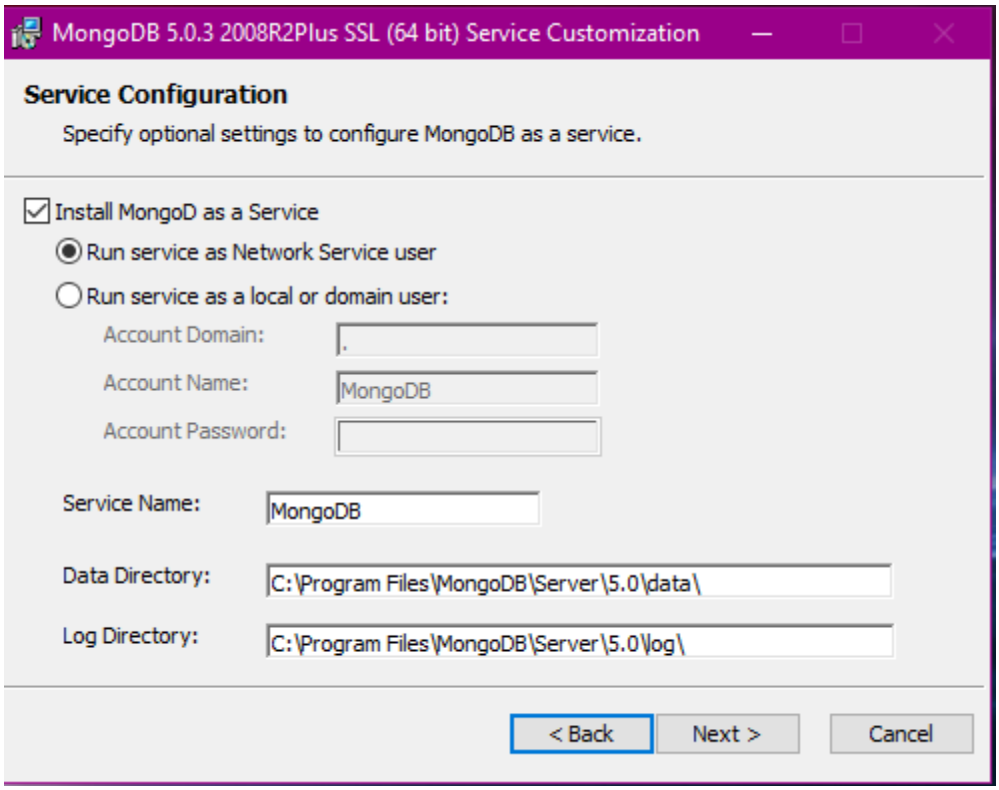


Elegimos hacer la instalación completa de MongoDB

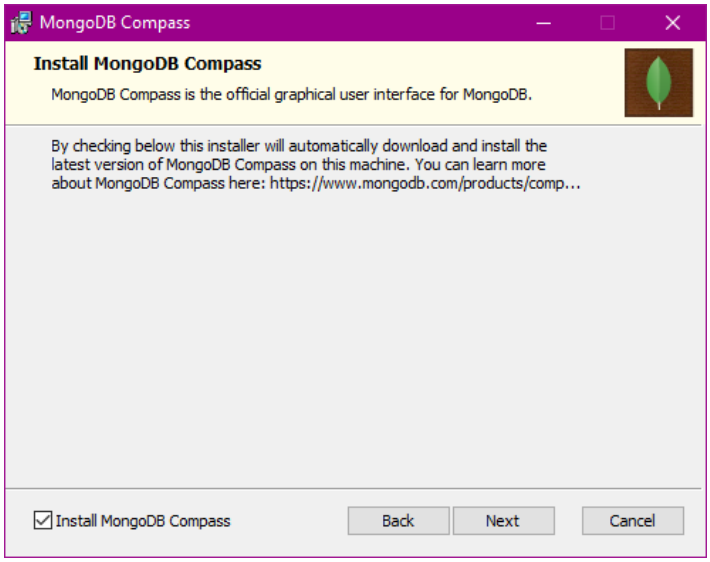
Antes de terminar la instalación, podemos comprobar que no hay ningún servicio relacionado con el mongo en el sistema:



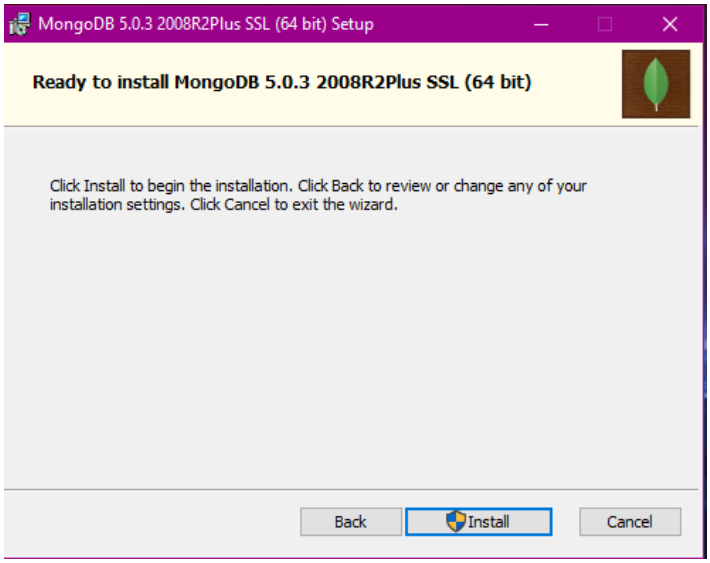
Vamos a seguir con la instalación y veremos cómo esto cambia:



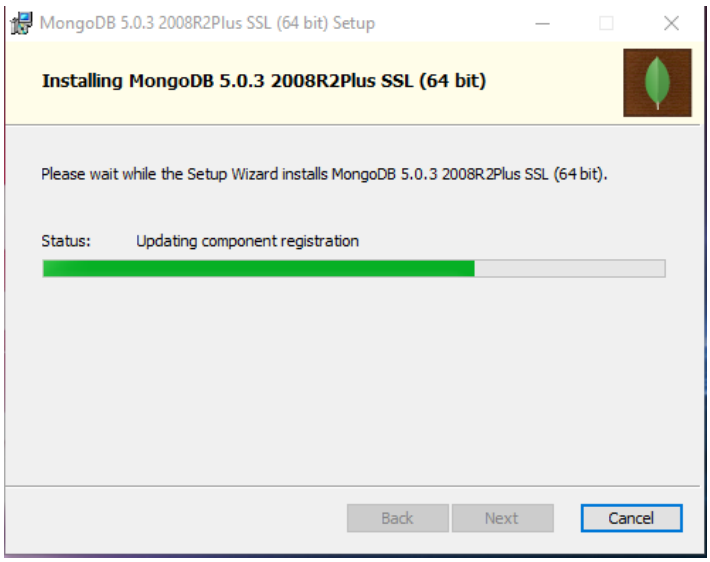
En esta ventana no tocamos nada, a no ser que queramos cambiar los directorios de datos (no es el caso). Pulsamos siguiente.



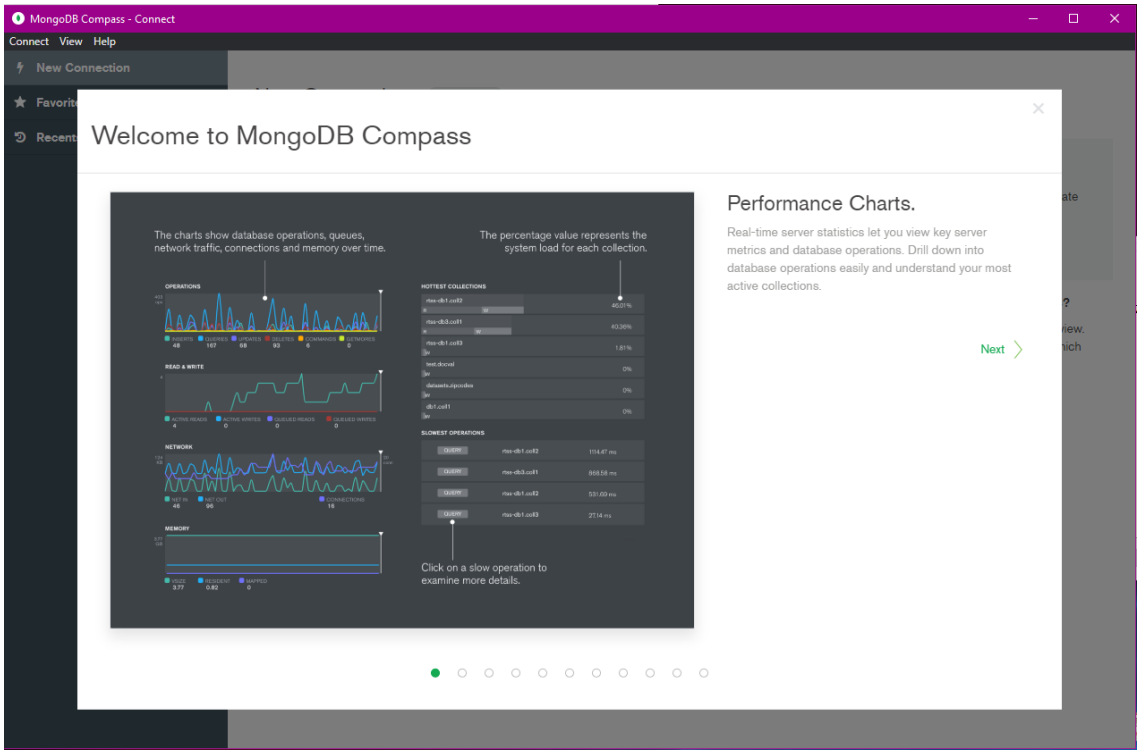
Siguiente.



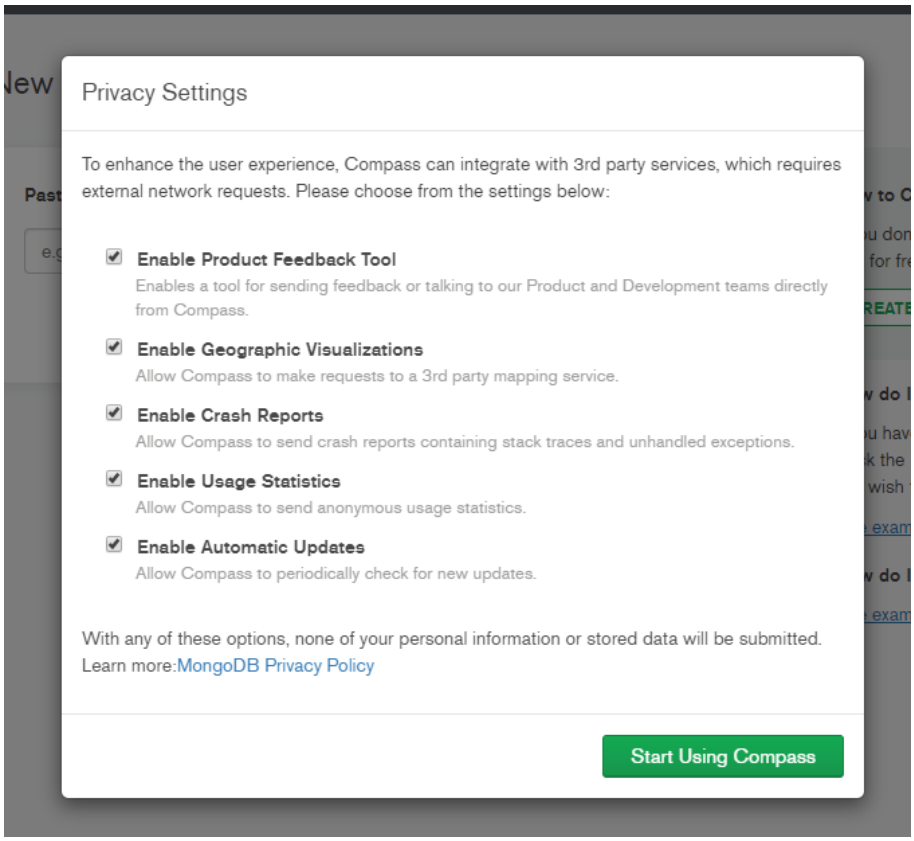
Y pulsamos en instalar. La instalación comenzará y podrá tomar unos minutos:



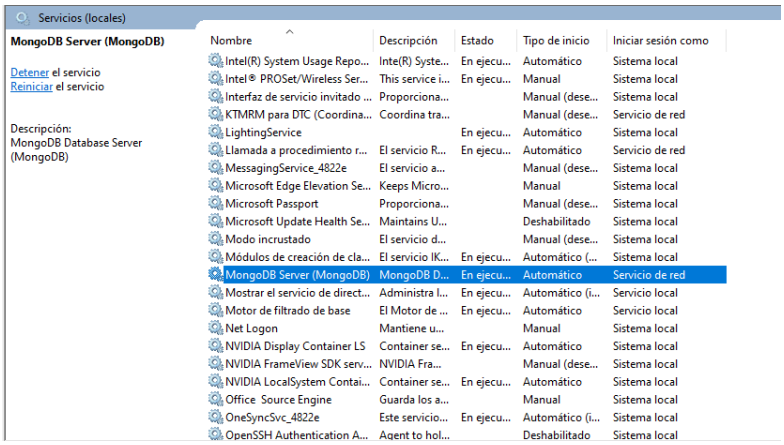
Una vez instalado, Mongo se abrirá y esta ventana será la primera que nos saldrá:



Cerramos esa ventana y nos saldrá otra en la que nos dirán que tenemos que aceptar los ajustes de privacidad:



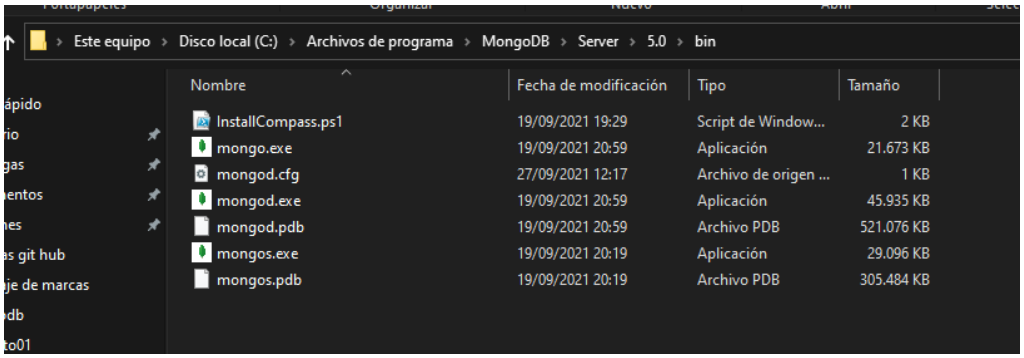
Ahora podemos entrar en los servicios del sistema y comprobar que se ha creado un servicio relacionado con el Mongo:



Si no queremos que este servicio consuma recursos del sistema, podemos deshabilitar el inicio automático de este servicio dándole clic derecho y propiedades. Si hacemos esto, cada vez que queramos usar el mongo deberemos activar el servicio manualmente.

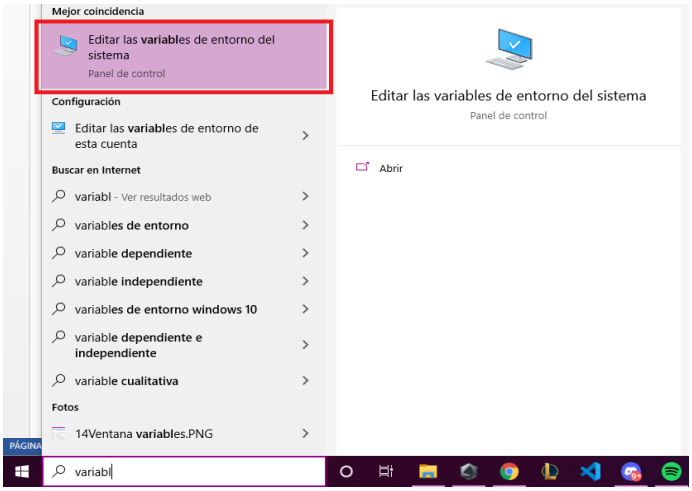
Ahora, para poder usar la Shell de mongo en la Powershell, deberemos añadir la ruta del mongo a la variable PATH, para ello, debemos hacer lo siguiente:

Primero tendremos que copiar la siguiente dirección, que es la ruta de trabajo de MongoDB:

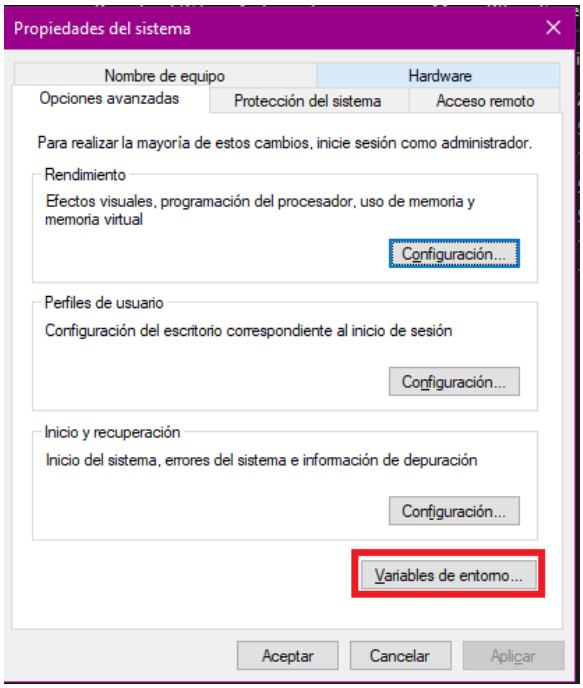


C:\Program Files\MongoDB\Server\5.0\bin

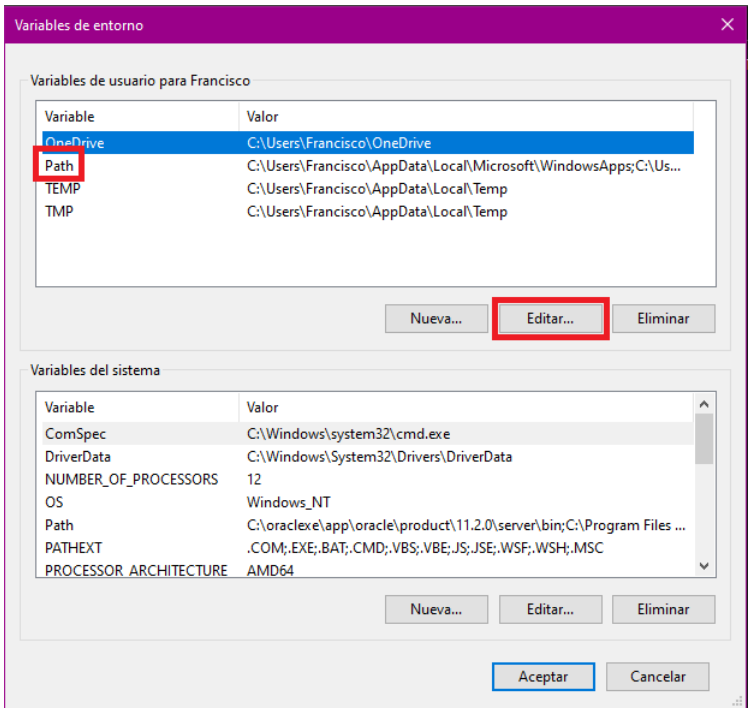
Una vez copiada la dirección, tenemos que irnos a las variables de entorno del sistema. Para ello buscaremos en el buscador de Windows “Variables de entorno”



Entramos y nos saldrá la siguiente ventana:

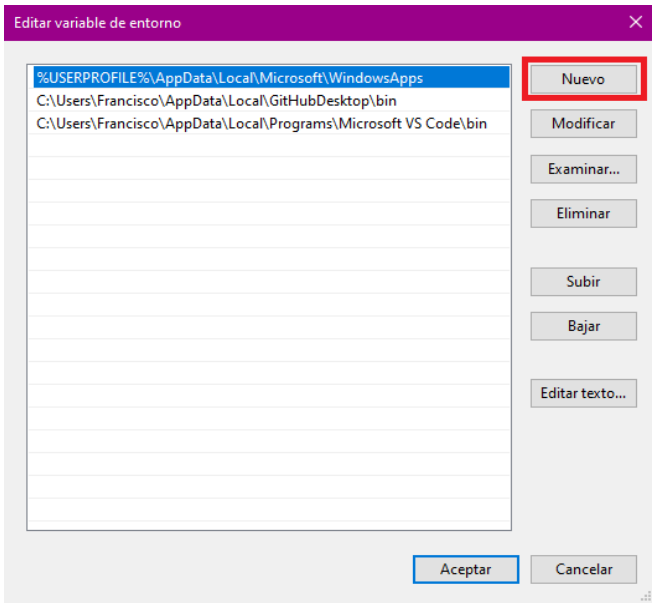


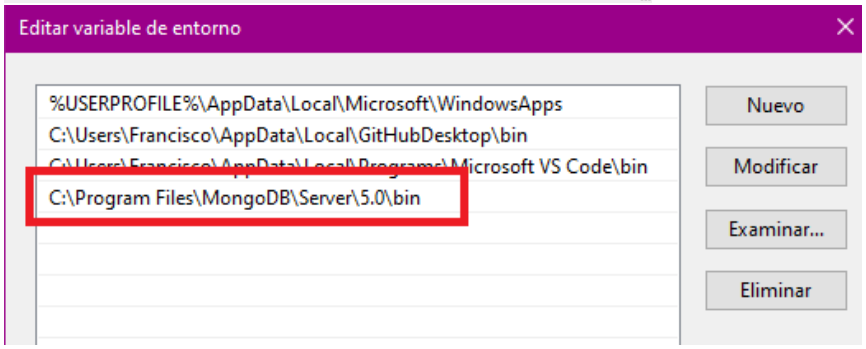
En esta ventana, pulsaremos el botón de variables de entorno:



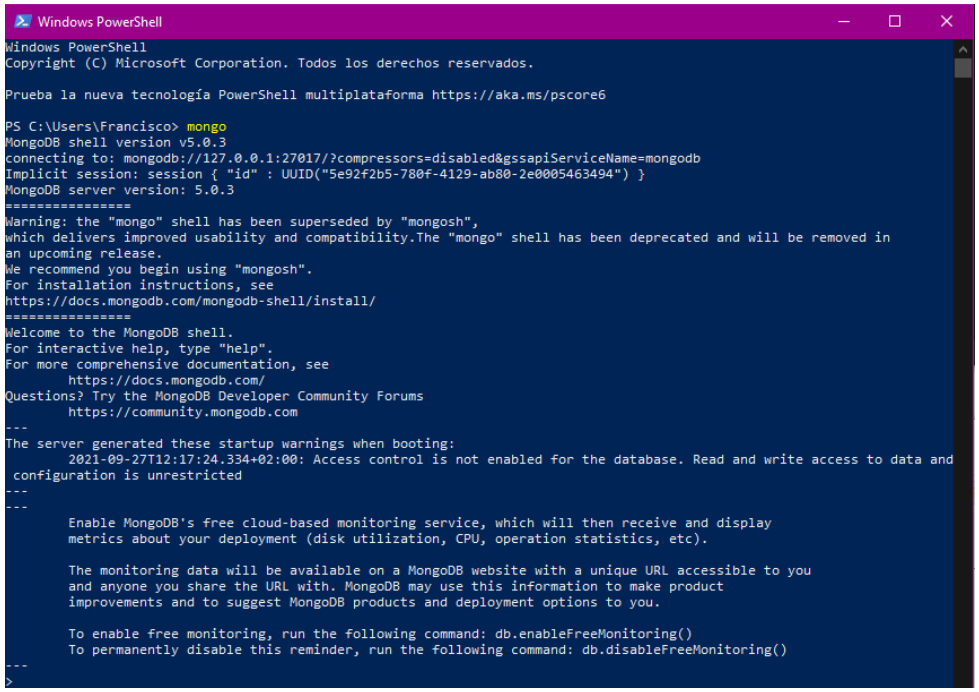
Seleccionamos la variables Path y le damos a editar.

En esta ventana, vamos a darle a nuevo y vamos a añadir la ruta que anteriormente copiamos:

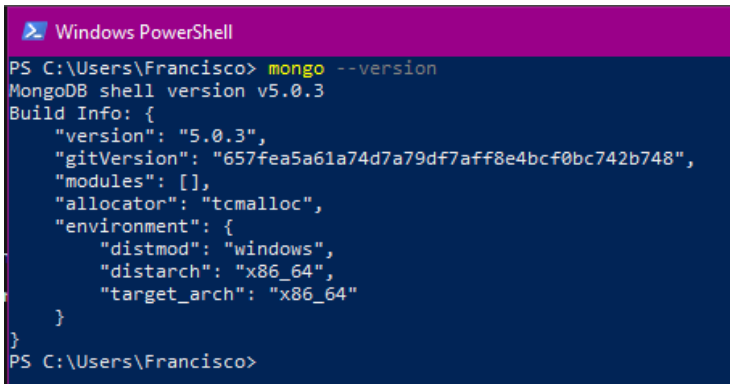




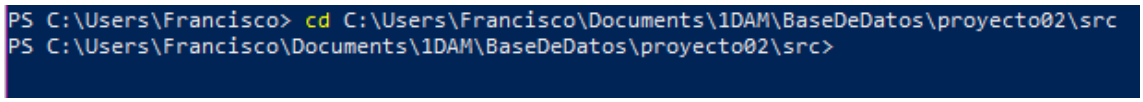
Una vez añadida la dirección a la variable Path, ya tenemos a nuestra disposición la Shell de Mongo en la PowerShell. Para comprobarlo, abriremos la PowerShell y ejecutaremos el comando “mongo”:



Como podemos comprobar, la Shell de mongo se abre correctamente. Podemos comprobar la versión con el comando “mongo --version”:

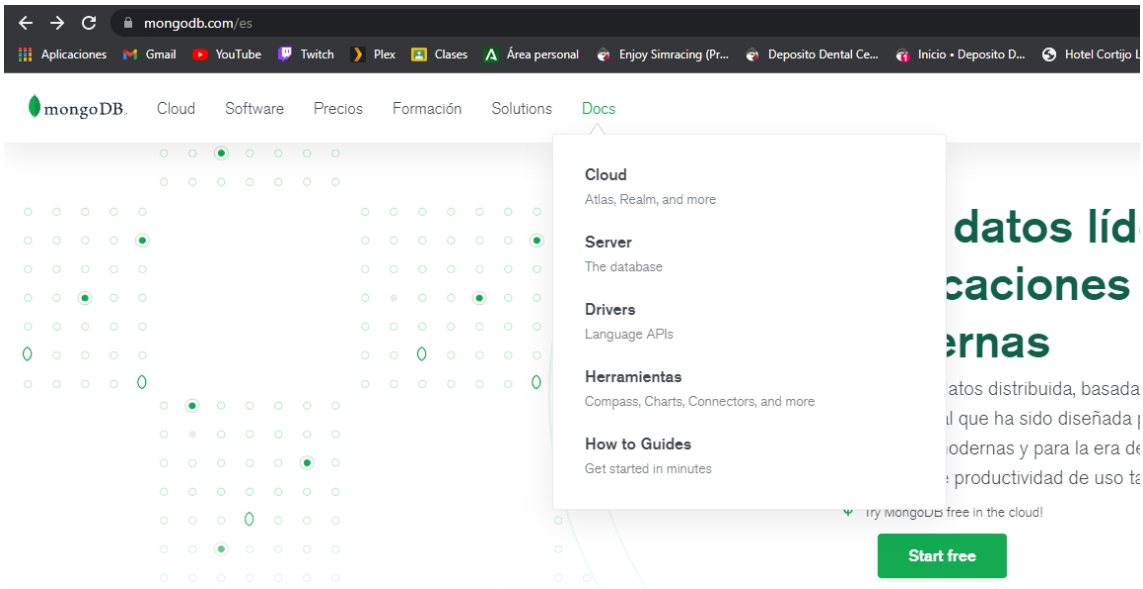


Desde aquí ya podemos ejecutar comandos de mongo y trabajar sobre él, pero antes de nada, vamos a salir con “exit” y vamos a irnos a otra carpeta para trabajar:



Y volvemos a entrar en mongo.

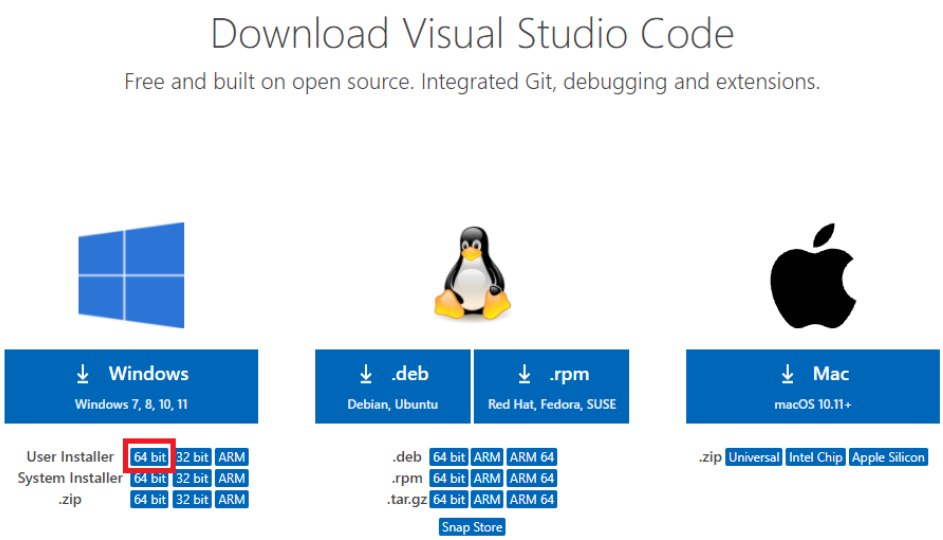
En este apartado de la web de Mongo podemos encontrar documentación importante:



4. Instalación Microsoft Visual Studio

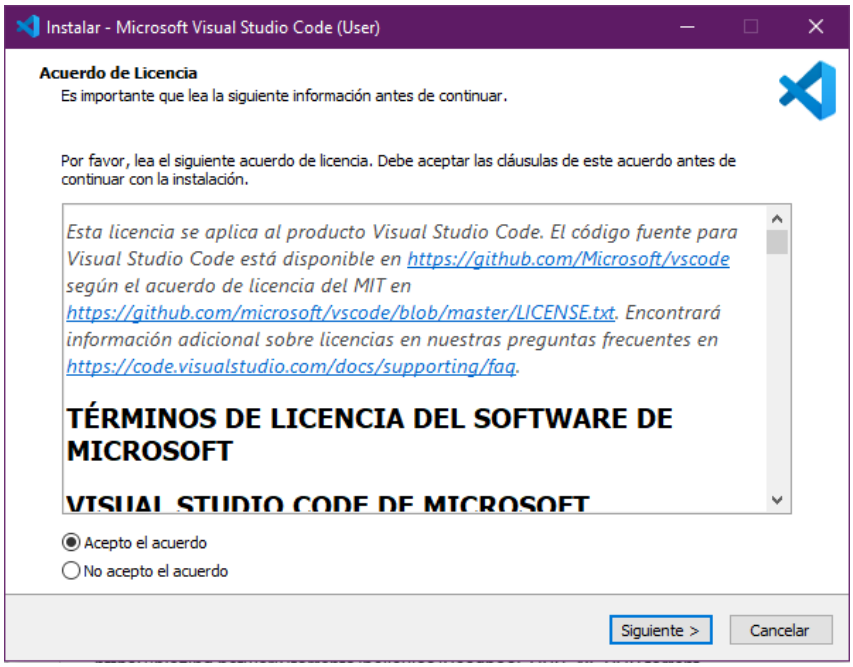
Para realizar la instalación de Visual Studio, tendremos que ir a la página oficial y descargarlo desde allí.

Una vez dentro, nos pedirá cuál es nuestro sistema operativo para proveernos una versión del programa hecha para este:

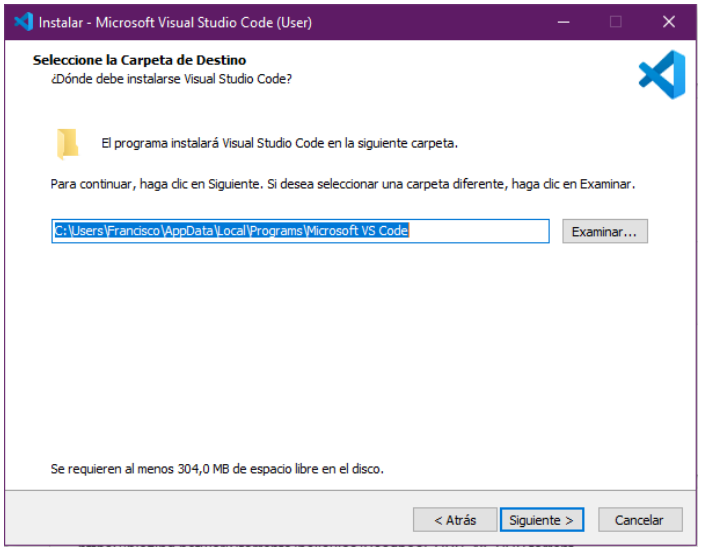


En mi caso voy a elegir Windows 64 bits.

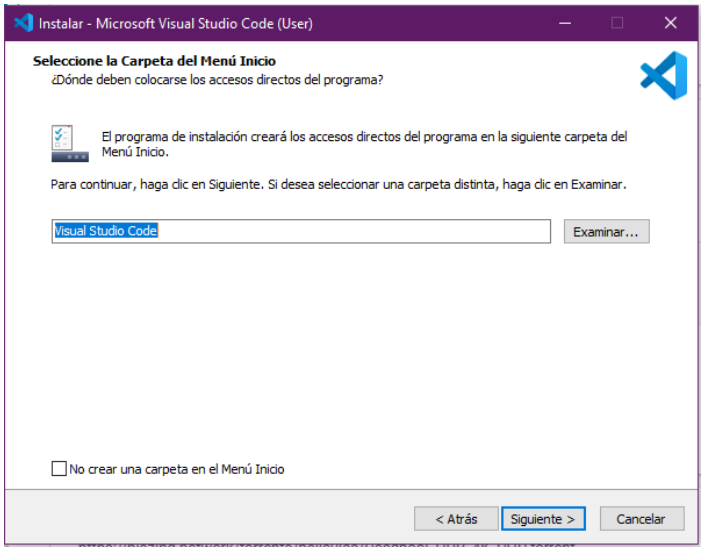
Una vez descargado, ejecutamos el .exe y nos saldrá el siguiente asistente:



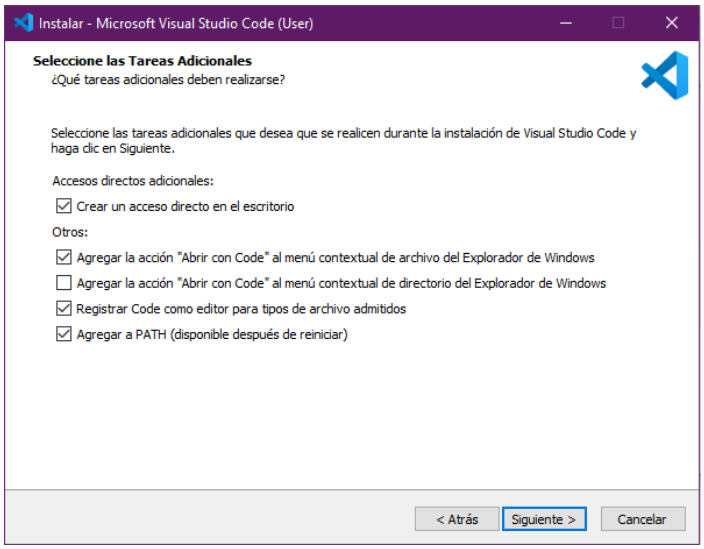
Aceptamos los términos y pulsamos siguiente



Elegimos la ruta de instalación, en este caso voy a elegir la que viene por defecto.

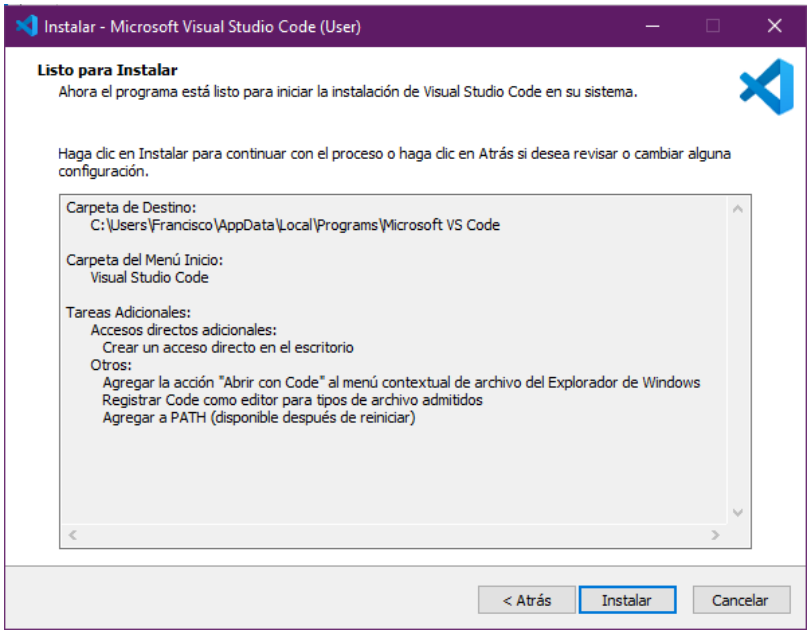


Siguiente

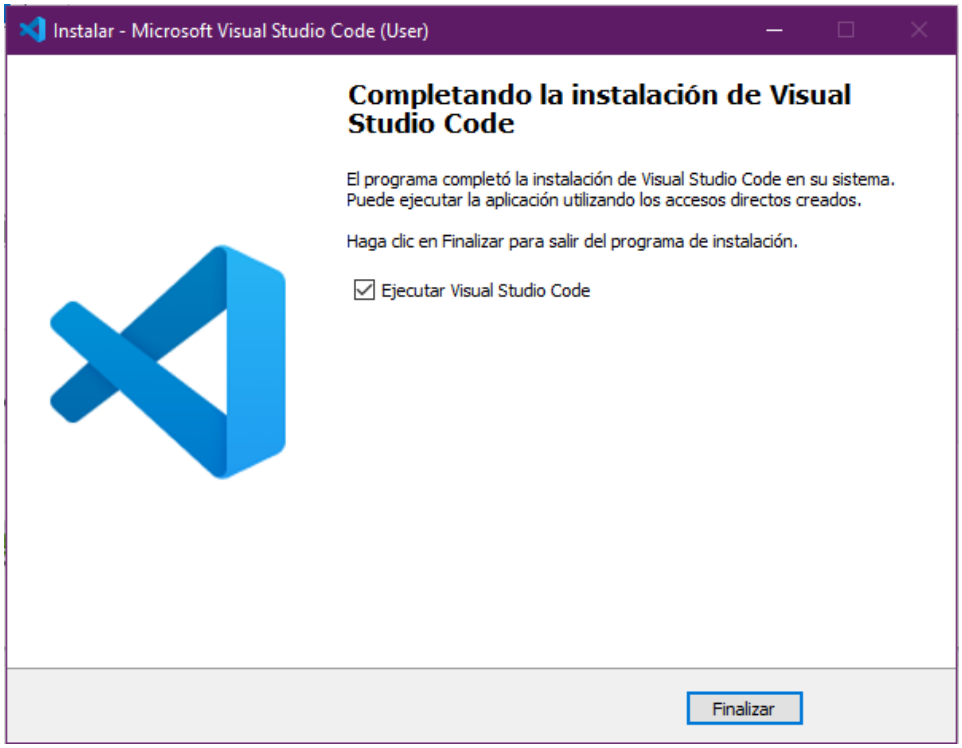


Marcamos las casillas que veamos convenientes.

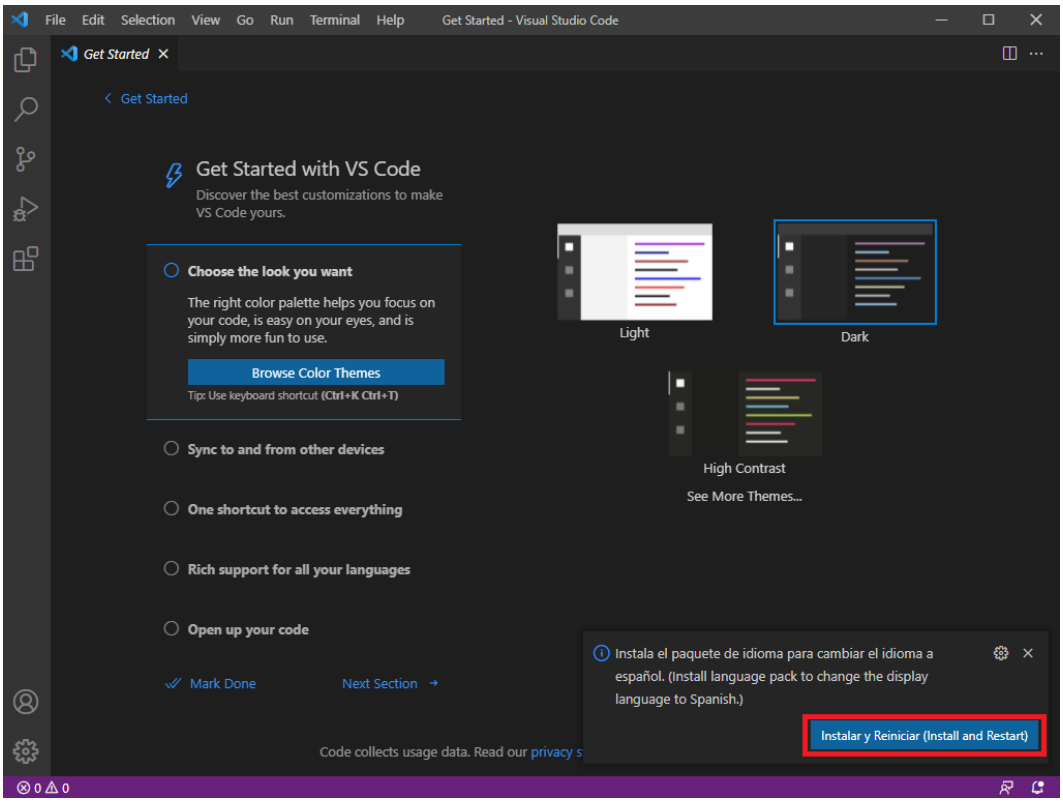
Y pulsamos en instalar:



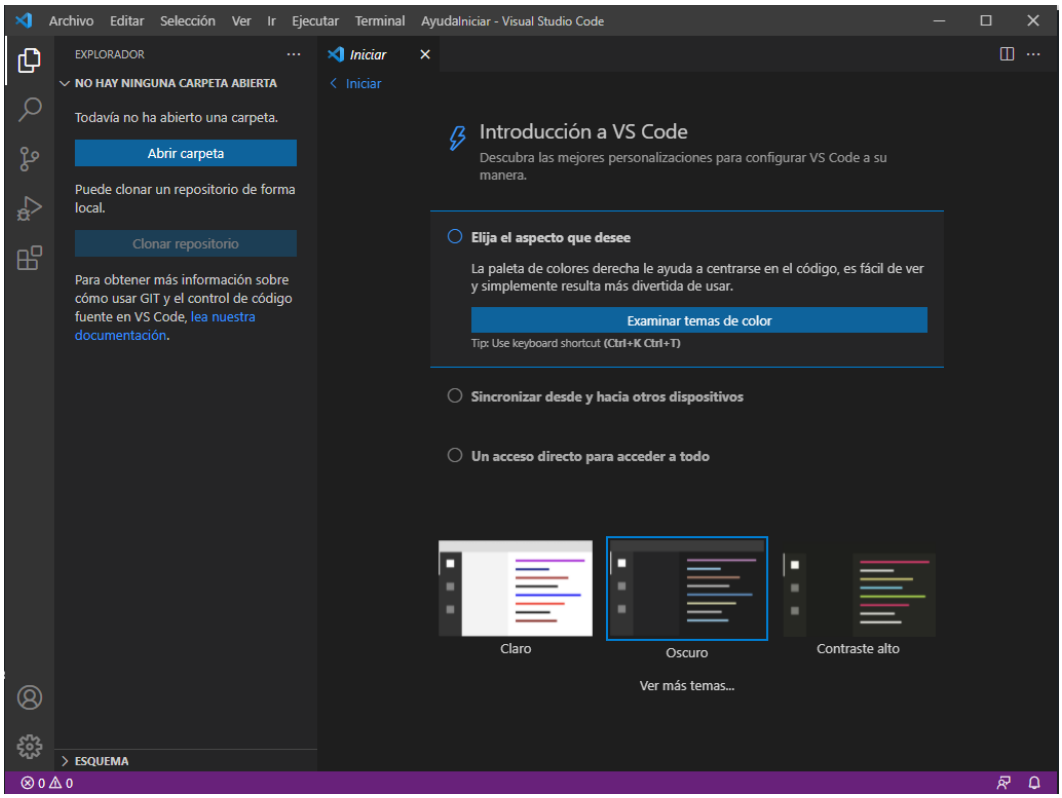
Una vez terminada la instalación, nos saldrá la siguiente ventana:



Ahora podemos ejecutar el Visual Studio pero, para tenerlo en español, nos saldrá un mensaje en la esquina inferior derecha que solicitará reiniciar el programa para aplicar la traducción. Le damos a reiniciar y al abrirse ya estará completamente en español:

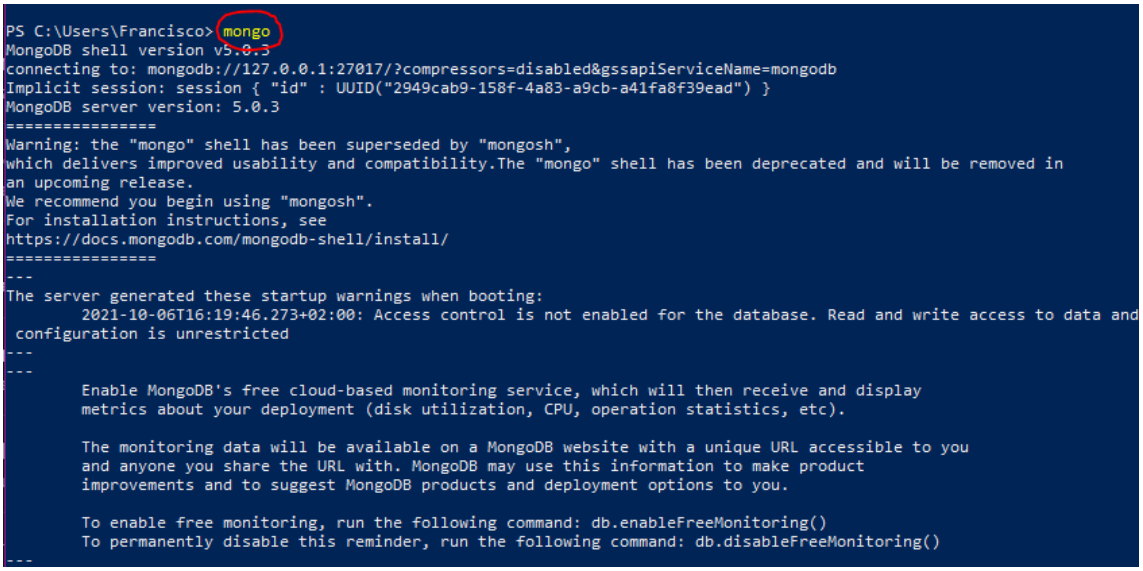


Visual Studio en español:

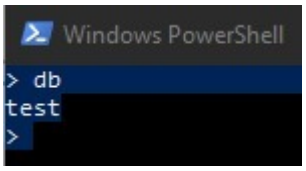


5. Uso de MongoDB

Para entrar en la consola de mongo, tenemos que poner el comando "mongo" en la PowerShell (se puede instalar una shell dedicada pero no vamos a hacerlo):

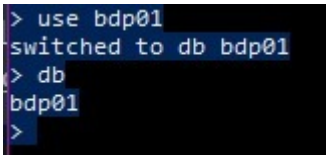


Con el comando "db" podremos mostrar sobre qué base de datos estamos trabajando:

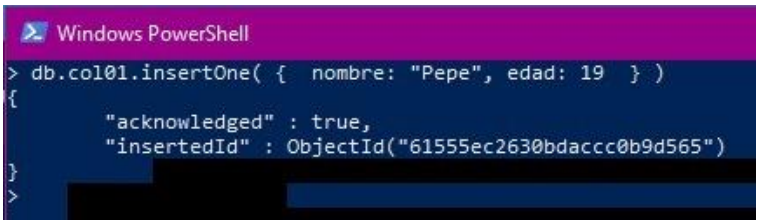


El gestor de base de datos tiene base de datos, como test, betis, etc, las bases de datos son "carpetas". Dentro de las carpetas tenemos colecciones (como si fueran hojas de cálculo). Cada colección tiene documentos .json, que serían las filas de la hoja de cálculo.

Con "use <database>" empezaremos a usar una base de datos (aunque no esté creada). Al insertar alguna información se creará la base de datos.



Con el comando "db.myCollection.insertOne({ x:1 });" insertamos información en la colección "myCollection" dentro de la base de datos que estamos usando actualmente:



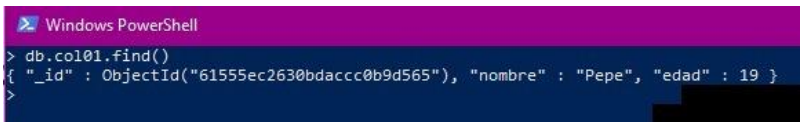
Gracias al ObjectId que le asigna mongo, podemos insertar el mismo campo tantas veces como queramos, que los distinguirá (podemos poner pepe 19 las veces que queramos).

“insertOne” permite 2 parámetros, aunque el “writeConcern” es opcional (no vamos a usarlo).

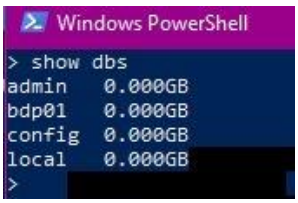
```
db.collection.insertOne(
    <document>,
    {
        writeConcern: <document>
    }
)
```

Dentro del parámetro "document" podemos escribir todos los datos que queramos.

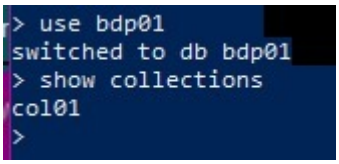
Con el comando "db.collection.find();" consultamos documentos en una colección:



El comando "show dbs" muestra todas nuestras bases de datos:

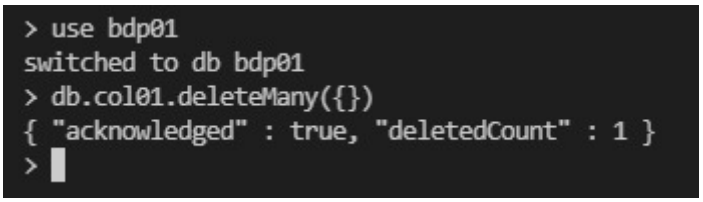


Con el comando "show collections" nos muestra las colecciones de las bases de datos actuales:



Con "db.collection.deleteMany" elimina datos de la colección.

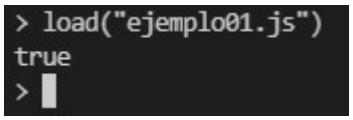
Con el campo ({}) vacío, elimina TODOS los documentos de la colección:



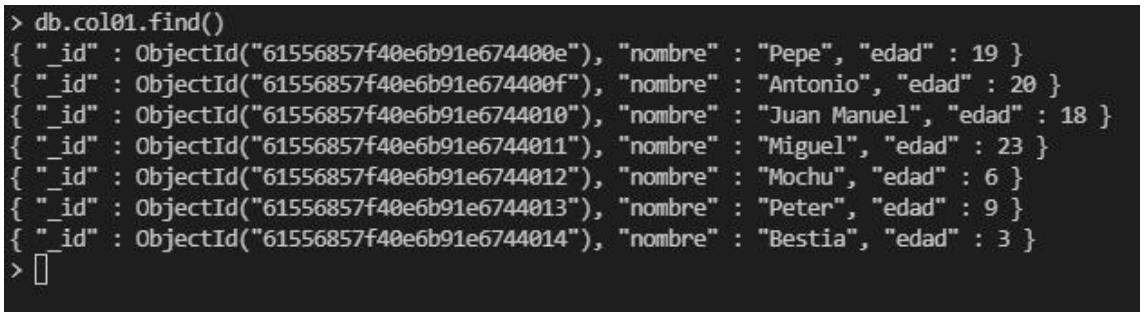
Miramos que se han eliminado:

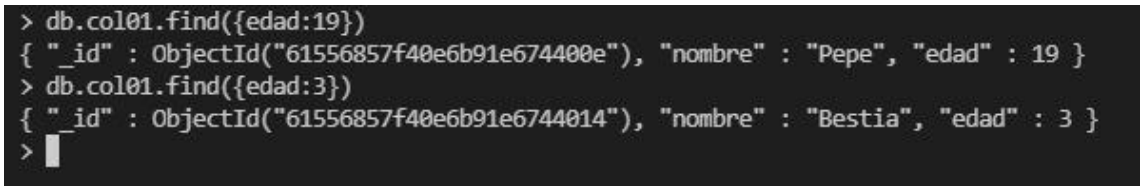
Ahora vamos a crear un script para lanzarlo desde la shell. Así siempre podremos volver a lanzarlo si tenemos algún problema. Este script vamos a crearlo desde Visual Studio Code, el cuál hemos instalado anteriormente:

Con *load("ejemplo01.js")* cargamos el script y se ejecuta dentro de mongo:



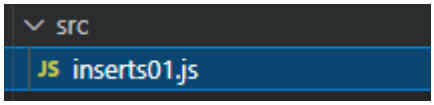
Si usamos “find" podemos ver como se han rellenado todos los documentos:



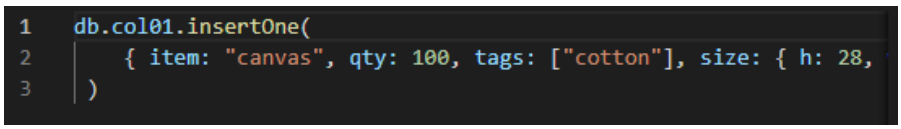


Insertar documentos

Vamos a trabajar desde la base de datos “bdp01”. Primero vamos a crear un fichero .js (inserts01.js) en la carpeta src:



Una vez creada vamos a insertar un documento en la colección “col01”:



Y ejecutamos el archivo .json desde la Shell (tenemos que estar en la misma ubicación que el fichero):

```
> load("inserts01.js")
true
>
```

Con find podemos ver que se ha cargado correctamente:

```
> db.col01.find()
{ "_id" : ObjectId("616c1a21e1d5f57b4bc6ea9c"), "item" : "canvas", "qty" : 100, "tags" :
 [ "cotton" ], "size" : { "h" : 28, "w" : 35.5, "uom" : "cm" } }
>
```

Si queremos insertar varios documentos en una misma orden, podemos usar el comando "insertMany", que inserta un array de documentos en la colección. Antes de ejecutar el comando, voy a usar un "deleteMany" para borrar los datos que hubiera de antes y dejar la colección limpia para hacer pruebas:

```
db.col01.deleteMany({})
db.col01.insertMany([
    { item: "journal", qty: 25, tags: ["blank", "red"], size: { h: 14, w: 21, uom: "cm" } },
    { item: "mat", qty: 85, tags: ["gray"], size: { h: 27.9, w: 35.5, uom: "cm" } },
    { item: "mousepad", qty: 25, tags: ["gel", "blue"], size: { h: 19, w: 22.85, uom: "cm" } }
])
```

Lo cargamos en la Shell:

```
> load("inserts01.js")
true
>
```

Y comprobamos que se ha cargado correctamente:

```
> db.col01.find()
{ "_id" : ObjectId("616c1d84e1d5f57b4bc6ea9d"), "item" : "journal", "qty" : 25, "tags" : [ "blank", "red" ], "size" : { "h" :
14, "w" : 21, "uom" : "cm" } }
{ "_id" : ObjectId("616c1d84e1d5f57b4bc6ea9e"), "item" : "mat", "qty" : 85, "tags" : [ "gray" ], "size" : { "h" : 27.9, "w" :
35.5, "uom" : "cm" } }
{ "_id" : ObjectId("616c1d84e1d5f57b4bc6ea9f"), "item" : "mousepad", "qty" : 25, "tags" : [ "gel", "blue" ], "size" : { "h" :
19, "w" : 22.85, "uom" : "cm" } }
>
```

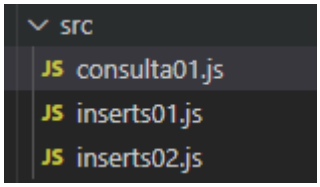
Ahora vamos a crear "Query documents" o documentos de consulta. Los voy a crear en otro archivo .js:

```
db.col01.deleteMany({})
db.col01.insertMany([
    { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
    { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },
    { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
    { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
    { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
]);
```

```
> load("inserts02.js")
true
>
```

```
> db.col01.find()
{ "_id" : ObjectId("616c243ee1d5f57b4bc6eaa0"), "item" : "journal", "qty" : 25, "size" : { "h" : 14, "w" : 21, "uom" : "cm" },
 "status" : "A" }
{ "_id" : ObjectId("616c243ee1d5f57b4bc6eaa1"), "item" : "notebook", "qty" : 50, "size" : { "h" : 8.5, "w" : 11, "uom" : "in"
}, "status" : "A" }
{ "_id" : ObjectId("616c243ee1d5f57b4bc6eaa2"), "item" : "paper", "qty" : 100, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" },
 "status" : "D" }
{ "_id" : ObjectId("616c243ee1d5f57b4bc6eaa3"), "item" : "planner", "qty" : 75, "size" : { "h" : 22.85, "w" : 30, "uom" : "cm"
 }, "status" : "D" }
{ "_id" : ObjectId("616c243ee1d5f57b4bc6eaa4"), "item" : "postcard", "qty" : 45, "size" : { "h" : 10, "w" : 15.25, "uom" : "cm
" }, "status" : "A" }
>
```

Este archivo lo hemos creado para hacer consultas. Para probarlo, vamos a crear un archivo llamado “consulta01.js” donde vamos a generar las consultas para después copiarlas y ejecutarlas en el terminal:



Voy a hacer pruebas de consultas con diferentes comandos:

```
db.col01.find( { item: "postcard" } )
```

Resultado:

```
> db.col01.find( { item: "postcard" } )
{ "_id" : ObjectId("616c243ee1d5f57b4bc6eaa4"), "item" : "postcard", "qty" : 45, "size" : { "h" : 10, "w" : 15.25, "uom" : "cm" }, "st
atus" : "A" }
>
```

Misma búsqueda con operador $eq:

```
db.col01.find( { item: { $eq: "postcard" } } )
```

Resultado:

```
> db.col01.find( { item: { $eq: "postcard" } } )
{ "_id" : ObjectId("616c243ee1d5f57b4bc6eaa4"), "item" : "postcard", "qty" : 45, "size" : { "h" : 10, "w" : 15.25, "uom" : "cm" }, "st
atus" : "A" }
>
```

Este comando muestra los campos en los que hay un valor qty igual a 25:

```
db.col01.find( {qty: { $eq: 25} } )
```

Resultado:

```
> db.col01.find( {qty: { $eq: 25} } )
{ "_id" : ObjectId("616c243ee1d5f57b4bc6eaa0"), "item" : "journal", "qty" : 25, "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "status
" : "A" }
>
```

Mongo también interpreta búsquedas dentro de arrays ya que es consciente de los arrays que hay.

Tenemos muchos más tipos de comparadores que podemos ver en la página oficial de mongo.

Aquí podemos ver algunos ejemplos:

`$eq`	Matches values that are equal to a specified value.
`$gt`	Matches values that are greater than a specified value.
`$gte`	Matches values that are greater than or equal to a specified value.
`$in`	Matches any of the values specified in an array.
`$lt`	Matches values that are less than a specified value.
`$lte`	Matches values that are less than or equal to a specified value.
`$ne`	Matches all values that are not equal to a specified value.
`$nin`	Matches none of the values specified in an array.